Forth-Methodik in der Programmierung

Eine freie Übersetzung des Textes *"Forth Methodology Applied to Programming"* vom *Jeff Fox* ins Deutsche.

1. **ANFANGEN - ANFANGEN – ANFANGEN**
   Erkenne und verwirf (deine) Illusionen und Ideen, die nicht hilfreich sind.
2. Identifiziere und verwerfe die Nichtprobleme, die dein Problem blockieren, oder behalte sie, aber trickse sie aus, damit sie helfen, das eigentliche Problem zu lösen.
3. Betrachte zuerst die Optionen, die alle anderen als erstes ablehnen würden, da hier wahrscheinlich die größte algorithmische Verbesserung zu finden ist.
4. Entwickele eine Lösung, die sorgfältig und gut durchdacht ist.
5. Optimiere diesen Lösungsentwurf so weit wie möglich in Richtung der Punkte 2, 3 und 4.
6. Erkenne und verwerfe die Illusion, daß ein Neuanfang schlecht ist, und du bereit bis zu programmieren. Kehre statt dessen mit einem besseren Verständnis der Probleme zu Punkt 1 zurück, **BIS** du alle guten Lösungen gesammelt hast, die du dir vorstellen kannst.
7. Vergleiche die Lösungen. Entwickele Experimente, Benchmarks, Simulationen oder was auch immer du benötigen, um zu bestätigen, daß du den optimalen Ansatz für das Problem gefunden hast.
8. Kehre zu Punkt 1 zurück, und zwar so lange, **BIS** die Ideen von Grund auf richtig sind ("right by design").
9. Nachdem das Problem durch den Ansatz *"Vor der Codieren: Faktorisieren! – Faktorisieren! – Faktorisieren!"* brutal einfach erscheint, besteht die triviale Programmieraufgabe darin, eine Eins-zu-eins-Abbildung der optimalen Lösung in Forth-Code zu konstruieren.
10. Der Forth-Code.
    - Erstelle eigene Werkzeuge (Tools), wenn sie hilfreich sind.
    - Schreibe Code, der so einfach und klar ist, dass Fehler einfach nicht passieren können.
    - Entwirf den Code von Grund auf richtig ("right by design").
    - **:** definiere Einzeiler von ungefähr dieser Länge **;**
    - Teste jedes Forth-Wort interaktiv.
    - Erweitere die Kernsprache zu einer eigenen Sprache (DSL) und nähere dich der Lösung an.
    - Kehre zurück zu Punkt 1, **BIS** schließlich die Code-Lösung herauskommt.
11. Schreibe die Dokumentation so einfach und klar, daß Fehler einfach nicht passieren können.
    - Dokumentiere die *"von Grund auf richtig machen"*-Algorithmen (*"right by design"*-Algorithmen) und den Code.
    - Erstelle ein Glossar mit einer Beschreibung für jedes eingeführte Forth-Wort.
    - Habe die ganze Zeit hindurch Spaß.
    - Genieße die Befriedigung über eine gut gemachte Arbeit.

Wenn man es richtig macht, ist Punkt 9 der einfachste Schritt.
Wenn man mit Punkt 9 beginnt, versteht man 1% dieser Methodik.

Diese Methodik, die hier eine Programmieraufgabe beschreibt, kann auf die Entwicklung von Hardware, Software und auf jede Art von Problem angewendet werden.

Der Erfinder von Forth, *Charles "Chuck" Moore*, sagt, daß Forth entwickelt wurde, um *"unlösbare" Probleme* in der Informatik zu "vermeiden". Er sagt, daß das *"wirkliche" Problem* dasjenige Problem ist, *das man lösen möchte*; und daß das Hinzufügen von unlösbaren Problemen [und anderen Balast] zu diesem wirklichen Problem das *eigentliche Problem* der Softwareentwicklung [und des Software-Engineerings] ist.

Mit jeder Iteration wird du das Problem besser verstehen, und jedes Mal, wenn du die obigen Schritte 1 bis 11 durchläufest, findest du verbleibende Stellen für weitere Optimierungen. Nachdem du den letzten Engpaß gefunden und beseitigt hast, suchst du mit jeder weiteren Iteration nach den nächstgrößeren Engpaß und beseitigst diesen.

Chuck Moore wiederholte die Schritte 1 bis 11 über 20 Jahre lang für Software. Er erklärte, daß das Problem der Programmierung gelöst sei. Das verbleibende Problem war die Hardware. Er untersuchte das Problem der Hardware und baute dazu seine eigenen neuen Werkzeuge.

Er wiederholte die Schritte 1 bis 11 für Hardware/Software weitere 20 Jahre lang. Computer sind Hardware/ Software und er kam der Lösung des Problems immer näher.

Er verwendete schrittweise, iterative Änderungen [und folgte dabei den Schritten 1 bis 11 seines Stufenmodells]. Er sagte, daß er auf einigen Stufen falsch abgebogen sei und zurückgehen mußte, um den Weg nach oben wiederzufinden. Lernen zu erkennen, wann du eine Stufe nach oben verlassen mußt. Ich bewundere ihn dafür, wie oft er uns gesagt hat, daß er sich in einem Ansatz oder in einer Schlußfolgerung geirrt hatte, daß er den falschen Weg eingeschlagen hatte, oder daß er uns etwas erzählt hatte, was nicht stimmt.

Ich sehe viele Leute, die einen Sprung wagen oder die senkrechte Wand hinaufklettern. Ich sage ihnen, daß es Treppen gibt, und daß das ein längerer, aber einfacherer und besserer Weg ist. Und manche Leute bleiben stecken, wenn sie versuchen, mit der Rolltreppe nach unten zu fahren. Erwäge die Treppe zu nehmen, [denn Einfachheit gewinnt ("simplicity wins")].

Wenn du irgendwo eine Rolltreppe nach oben findest, laß es uns alle wissen!

Jeff Fox, 09.12.2001 (https://www.ultratechnology.com/method.htm)